УДК 004.652.43, 004.421.4 DOI: 10.14529/cmse190206

ОБНОВЛЕНИЕ МНОГОТАБЛИЧНЫХ ПРЕДСТАВЛЕНИЙ НА ОСНОВЕ КОММУТАТИВНЫХ ПРЕОБРАЗОВАНИЙ БАЗЫ ДАННЫХ

© 2019 В.С. Зыкин[1], М.Л. Цымблер[2]

[1] *Омский государственный технический университет*
(644050 Омск, пр. Мира, д. 11),
[2] *Южно-Уральский государственный университет*
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: vszykin@mail.ru, mzym@susu.ru

Поступила в редакцию: 26.09.2018

В современных технологиях реляционных баз данных механизм представлений (view) реализует внешний уровень архитектуры ANSI–SPARC, скрывая детали концептуальной структуры базы данных от конечных пользователей. Однако использование данного механизма сопряжено с необходимостью решения задачи корректного обновления представлений: СУБД должна обеспечить корректное выполнение операций вставки, удаления или обновления кортежа в представлении над соответствующими базовыми отношениями данного представления. Для решения указанной задачи в стандарте SQL вводится жесткое ограничение: модифицируемому кортежу представления может соответствовать только один кортеж в базовом отношении. Триггеры, реализующие обновление представлений, обладают рядом недостатков: необходимость создания триггера для каждого представления базы данных, непредсказуемый порядок запуска триггеров, относящихся к одному представлению и др. В статье рассматривается подход к решению данной задачи на основе применения коммутативных преобразований базы данных. При этом не накладывается ограничение единственности кортежа базового отношения, соответствующего обновляемому кортежу в представлении. Описан Сопроцессор СУБД, который размещается на клиентском компьютере и обеспечивает коммутативные преобразования в отношениях базы данных, хранимых на сервере. Сопроцессор выполняет формирование текста транзакции, реализующей коммутативные преобразования, и осуществляет запуск этой транзакции на сервере. Представлена реализация сопроцессора для свободной СУБД PostgreSQL. Проведены вычислительные эксперименты, подтверждающие эффективность предложенного подхода в приложениях классов OLAP и OLTP.

Ключевые слова: коммутативное преобразование, реляционная алгебра, многотабличное представление, обновление представлений, реляционная СУБД, триггер.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Зыкин В.С., Цымблер М.Л. Обновление многотабличных представлений на основе коммутативных преобразований базы данных // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 2. С. 92–106. DOI: 10.14529/cmse190206.

Введение

Современные системы баз данных строятся в соответствии с трехуровневой архитектурой ANSI—SPARC [11]. Внутренний (физический) уровень отвечает за физический способ организации данных. Промежуточный (концептуальный) уровень инкапсулирует реляционную схему базы данных. Внешний (пользовательский) уровень показывает, как выглядит база данных с точки зрения конечного пользователя и реализуется с помощью представлений. *Представление (view)* — это виртуальное (логическое) отношение базы данных, которое является синонимом запроса к хранимым отношениям базы данных. Механизм представлений позволяет скрывать детали концептуальной структуры базы данных от конечных пользователей. Однако

использование данного механизма сопряжено с необходимостью решения задачи корректного *обновления представлений*, которая заключается в следующем. Поскольку представление воспринимается конечным пользователем как хранимое отношение базы данных, то возможны операции вставки, удаления или обновления кортежа в представлении. СУБД должна обеспечить корректное выполнение указанных операций над соответствующими базовыми отношениями данного представления.

В стандартах языка баз данных SQL, начиная с первой версии [12], решение данной задачи основано на введении ограничений на структуру представления. В соответствии с этими ограничениями для каждого кортежа представления необходимо наличие соответствующего кортежа в базовом отношении, а для каждого обновляемого атрибута представления необходимо наличие соответствующего ему атрибута в базовом отношении. В стандарте SQL:1999 вводится концепция триггера, существенно расширяющая функциональные возможности СУБД [20]. *Триггер* представляет собой подпрограмму на процедурном расширении SQL, постоянно хранимую в виде исходного текста в базе данных, и ассоциированную со специфицированным событием в базе данных. При наступлении указанного события СУБД автоматически исполняет тело триггера. Триггеры могут быть использованы для обновления представлений: с событием обновления представления необходимо ассоциировать тело триггера, выполняющее надлежащее обновление базового отношения.

Однако по следующим причинам триггеры не являются идеальным решением рассматриваемой задачи. Триггеры, реализующие обновление представлений, требуется разрабатывать для каждой отдельной базы данных. Триггеры являются источником накладных расходов СУБД, поскольку требуют блокировки различных ресурсов, проверки наступления события триггера, поддержки временных таблиц (например, таблицы INSERTED и DELETED в СУБД MS SQL Server используются для проверки результатов изменений данных и установки условий срабатывания триггеров [8]) и др. Кроме того, при наличии нескольких триггеров, относящихся к одному отношению (представлению) и ассоциированных с одним событием, последовательность их запуска не детерминирована.

Научные исследования, посвященные задаче корректного обновления представлений, начаты с момента становления теории реляционных баз данных и продолжаются в настоящее время. Однако, как показывает обзор научных публикаций по данной тематике, универсальное решение пока не найдено.

В данной работе предлагается подход к обновлению представлений, основанный на аппарате коммутативных преобразований данных. Коммутативность операций преобразования представления и преобразования отношений понимается как соответствие начального и конечного состояний базы данных между различными преобразованиями. Обновление представления осуществляется с помощью сопроцессора коммутативных преобразований, который создает транзакцию, выполняющую коммутативные преобразования отношений базы данных.

Статья организована следующим образом. В разделе 1 приводится обзор работ по теме исследования. Раздел 2 кратко описывает аппарат коммутативных преобразований. Приводятся формулы реляционной алгебры для операций удаления, добавления и обновления кортежа в представлении. В разделе 3 описана архитектура и принципы реализации сопроцессора коммутативных преобразований. В разделе 4 представлены

результаты вычислительных экспериментов, исследующих эффективность предложенного подхода. В заключении резюмируются итоги выполненного исследования.

1. Обзор работ

Научные исследования, посвященные задаче корректного обновления представлений, начаты с момента становления теории реляционных баз данных и продолжаются в настоящее время.

Работу 1981 г. [4], в которой предложено понятие коммутативных преобразований и сформулированы условия коммутативности обновления представлений, можно назвать пионерской в рассматриваемой области исследований. Авторы, однако, не предлагают алгоритмов обновления представлений.

В статье 1982 г. Даял исследует задачу соответствия модификаций пользовательских представлений данных с соответствующими преобразованиями в исходной базе данных [6]. В первую очередь в данной работе была рассмотрена корректность модификаций, их свойства и условия существования. Однако, практической реализации обновления представлений не приводится.

В 1984 г. Мазунага в работе [18] предложил вероятностный подход к формированию правил преобразования данных. Вероятностный характер результата преобразований является следствием использования семантического подхода. При возникновении нештатных ситуаций пользователю предоставляется возможность вручную откорректировать операции модификации. Очевидно, что в общем случае такой подход не гарантирует корректность результата обновления.

В 1985 г. Келлер в работе [14] предложил подход к решению задачи корректного обновления многотабличных представлений, который требует наличия в представлении атрибутов первичных ключей. Между тем в современной практике проектирования реляционных баз данных используются суррогатные первичные ключи отношений (вместо одного из атрибутов, выбираемого из множества потенциальных ключей), реализуемые в СУБД с помощью автоинкрементного типа данных. Суррогатные ключи, таким образом, не имеют определенной семантики и лишены возможности изменения.

В 1987 г. была разработана первая версия стандарта ISO/IEC 9075 [12], описывающего язык баз данных SQL. В данном стандарте присутствуют ограничения на обновления представлений, согласно которым один кортеж в представлении должн соответствовать одному кортежу в базовом отношении.

В 1988 г. Готтлоб и др. в работе [9] предложили понятие согласованного представления (consistent view), которое обладает следующим свойством: если возможны операции обновления представления, тогда имеется однозначный набор операций обновления базы данных, дающий тот же результат. Рассмотрена задача трансляции набора операций обновления представлений в набор операций обновления базы данных. Однако, в данной работе для кортежа согласованного представления допускается только один соответствующий кортеж в базовом отношении.

В 1990 г. Лангерак [15] предложил понятие репрезентативного экземпляра представления, который формируется из множества атрибутов всех отношений базы данных. Представления, таким образом, являются проекцией репрезентативного экземпляра. В данной работе, однако, рассмотрен только частный случай схемы баз данных,

когда каждое отношение является подмножеством проекций репрезентативного экземпляра по атрибутам каждого отношения.

В 2003 г. Лечтенбёргер в статье [16] представил подход «постоянного дополнения» (constant complement approach), развивающий идеи работы [4]. Данный подход предоставляет пользователю возможность отменить произведенные обновления представлений, используя результаты последующих обновлений. Практическая реализация предложенного подхода, однако, не описана.

В современной работе [5] Бертосси и Салими рассматривают задачу обновления представлений применительно к подготовке данных для машинного обучения. Удаление кортежей из представления рассматривается как исключение зашумленных данных из обучающей выборки. Таким образом, авторы ограничиваются рассмотрением только одной операции удаления кортежа из представления.

Рассмотренные работы подразумевают наличие ограничения, которое требует соответствия одного кортежа представления одному кортежу в базовом отношении, и рассматривают частные случаи обработки многотабличных представлений. В следующих разделах данной статьи мы рассмотрим подход, позволяющий преодолеть указанное ограничение в общем случае на основе использования аппарата коммутативных преобразований.

2. Коммутативные преобразования реляционных отношений

В данном разделе кратко представлен аппарат коммутативных преобразований [3], который является основой предлагаемого подхода к решению проблемы корректного обновления многотабличных представлений.

2.1. Базовые обозначения и определения

Далее используются следующие стандартные обозначения реляционных операций [7]: $\pi_X(R)$ — *операция проекции* отношения R по множеству атрибутов X, $\sigma_F(R)$ — *операция выбора* над отношением R, F — логическое выражение на значениях атрибутов, $R_1 \bowtie R_2$ — операция *естественного соединения* отношений R_1 и R_2.

В соответствии со статьей [3] введем обозначение модели информационной системы: $\Omega := (M, D, O, P)$, где M — описание схемы данных, D — множество допустимых состояний базы данных, O — множество операций модификации представлений, P — множество ограничений (предикатов) на состояния представления данных.

Введем краткую запись для последовательности операций естественного соединения нескольких отношений $R_1, \ldots, R_m$:

$$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_m := \bowtie_{i=1}^{m} R_i. \tag{1}$$

Определим *многотабличное представление данных* Q как результат запроса на выборку данных из отношений $R_1, \ldots, R_m$:

$$Q := \pi_{X_0}\Big(\sigma_F\big(\bowtie_{i=1}^{m} \pi_{X_i}(R_i)\big)\Big), \tag{2}$$

где X_0 — множества атрибутов, которые будут формировать заголовок результирующего отношения. X_i — подмножество атрибутов отношения R_i, которое будет использоваться для построения многотабличного представления. Множество атрибутов, задействованных в объекте $\mathcal{O}$, будем обозначать как $\langle\mathcal{O}\rangle$.

Далее определим, по каким атрибутам будет производиться операция проекции над отношениями. Пусть атрибут $A \in \langle R_i \rangle$, тогда $\forall i\ (1 \leqslant i \leqslant m) : A \in X_i$, если выполняется хотя бы одно из следующих условий:

1) $A \in X_0$;
2) $\exists R_\ell$: $A \in \langle R_\ell \rangle$, $\ell \neq i$;
3) $A \in \langle F \rangle$.

Для корректного выполнения операций коммутативных преобразований на схему базы данных M накладывается требование ее ацикличности. Совокупность отношений базы данных $R_1, R_2, \ldots, R_k$ является ацикличной, если отсутствуют упорядоченные транзитивные связи по внешним ключам между отношениями R_1 и R_2, между R_2 и R_3, $\ldots$, между R_{k-1} и R_k, а также связь между отношениями R_k и R_1 [2].

Пусть модель Ω будет *исходной*, а $\Omega' := (M', D', O', P')$ — *целевой*. Межмодельные преобразования подразумевают построение отображения $\Omega \Leftrightarrow \Omega'$. В данной работе показывается, каким образом отображение $\Omega \Rightarrow \Omega'$ может быть сведено к выполнению транзакции в базе данных.

Совокупность всех значений данных в базе данных, неизменных в течение некоторого промежутка времени, будем называть *состоянием базы данных*. Пусть при работе с моделью Ω' пользователь выполняет команду обновления представления f'_p, которая переводит модель Ω' из состояния $d'_i \in D'$ в состояние $d'_j \in D'$. Тогда необходимо выполнить преобразования, соответствующие переходу модели Ω из состояния $d_i \in D$ в состояние $d_j \in D$.

Преобразование представления данных, соответствующие отображению $\Omega \Leftrightarrow \Omega'$, будем считать *корректным*, если выполнены следующие *условия коммутативности*:

$$\begin{aligned} d_i &\xrightarrow{Q} d'_i \xrightarrow{f_p} d'_j, \\ d_i &\xrightarrow{Alg_p} d_j \xrightarrow{Q} d'_j, \end{aligned} \tag{3}$$

где Q — многотабличный запрос; Alg_p — алгоритм преобразования исходной базы данных, сопоставленный команде обновления f_p.

Таким образом, переход в состояние d'_j возможен двумя способами, но результат должен быть один и тот же. В работе [3] построены выражения реляционной алгебры для алгоритмов Alg_p, реализующих операции обновления многотабличных представлений.

Отношения $R_1, \ldots, R_m$ должны иметь упорядочение по внешним ключам: главные (ссылающиеся) отношения в этой последовательности стоят раньше, подчиненные (ссылаемые) — позже. Таким образом, существует частичный порядок, при котором имеется только одно отношение R_m, не имеющее подчиненных отношений. Далее это отношение будем называть *целевым*. Операции обновления данных должны быть реализованы только в целевом отношении R_m. Остальные отношения, используемые для формирования представления Q, назовем *отношениями-справочниками*.

2.2. Операции модификации многотабличного представления

В данном разделе приведена реализация реляционных операций обновления многотабличного представления на основе коммутативных преобразований данных. Операция обновления подразумевает одно из следующих действий, связанных с обновлением целевого отношения: удаление кортежа из Q, вставка кортежа в Q и

обновление кортежа в Q. Коммутативность операций удаления и вставки кортежа доказана в работе [3], суперпозиция указанных операций реализует операцию обновления кортежа.

Удаление кортежа

Допустим, что кортеж u удаляется из представления, соответствующего запросу Q (далее запрос и соответствующее ему отношение мы обозначаем как Q). Результат данной операции можно выразить функцией $Q' := DELETE(Q, u)$, где в качестве параметров фигурируют удаляемый кортеж u и многотабличное представление Q, а возвращаемым значением является представление Q':

$$DELETE(Q, u) := R_m \setminus \pi_{\langle R_m \rangle}(T_D), \tag{4}$$

где T_D — множество кортежей в целевом отношении R_m, совпадающих с кортежем u на атрибутах X_0. Множество T_D определяется следующим образом:

$$\begin{gathered} T_D := \pi_{X_0}\Big(\sigma_{c_{del}}\big(\bowtie_{i=1}^{m} \ \pi_{X_i}(R_i)\big)\Big), \\ c_{del} := F \wedge (X_0 = u), \end{gathered} \tag{5}$$

где запись $X_0 = u$ означает равенство значений атрибутов кортежа, удаляемого из представления, и соответствующих атрибутов в отношениях базы данных. Во время выполнения операции удаления кортежей в соответствии с семантикой предметной области формируется T_D — множество кортежей целевого отношения, соответствующих удаляемому кортежу в представлении.

Вставка кортежа

Пусть выполняется вставка кортежа u в представление Q. Представим результат данной операции в виде функции $Q' := INSERT(Q, u)$, где Q — многотабличное представление, u — новый кортеж, Q' — возвращаемое многотабличное представление со вставленным кортежем. При выполнении этой функции в целевое отношение R_m необходимо вставить множество кортежей T_I.

$$INSERT(Q, u) := R_m \cup T_I, \tag{6}$$

где множество кортежей T_I для вставки в целевое отношение определяется следующим образом:

$$\begin{aligned} T_I &:= \pi_{X_m}\big(\sigma_F(T'_I \bowtie u)\big), \\ T'_I &:= \pi_Y\Big(\sigma_{c_{ins}}\big(\bowtie_{i=1}^{m-1} \ \pi_{X_i}(R_i)\big)\Big), \\ Z &:= X_0 \cap \cup_{i=1}^{m-1} X_i, \\ Y &:= \Big(\langle R_m \rangle \cup \langle F \rangle \cup X_0\Big) \cap \cup_{i=1}^{m-1} X_i, \\ c_{ins} &:= F' \wedge (Z = u[Z]). \end{aligned} \tag{7}$$

Здесь F' представляет собой набор условий, основанный на множестве F, и не включающий в себя условия, накладываемые на целевое отношение: $\langle F' \rangle := \langle F \rangle \cap \cup_{i=1}^{m-1} X_i$ [19].

В процессе формирования множества кортежей T_I, формируется представление T'_I. Заголовок представления T'_I состоит из атрибутов всех отношений, за исключением целевого. Представление T'_I можно интерпретировать как множество всех кортежей базы данных, которые могут быть связаны с кортежем, добавляемым в представление Q («множество универсум»). Далее для формирования представления T_I выполняются следующие действия. Множество T'_I с помощью операции естественного соединения связывается с кортежем u, затем применяются следующие операции: выбор по условиям F и проекция по атрибутам целевого отношения.

Обновление кортежа

Пусть в представлении Q кортеж u заменяется на кортеж u', тогда обновление представления выражается функцией Q: $Q' := UPDATE(Q, u, u')$. Данную функцию можно представить в виде суперпозиции функций для операций удаления текущего кортежа и добавления нового кортежа:

$$UPDATE\big(Q, u, u'\big) := INSERT\Big(DELETE\big(Q, u\big), u'\Big). \tag{8}$$

3. Сопроцессор коммутативных преобразований

В данном разделе описан Сопроцессор СУБД, который обеспечивает коммутативные преобразования в базе данных при обновлении многотабличного представления.

3.1. Архитектура и методы реализации

Сопроцессор коммутативных преобразований (СКоП) — это программная система, которая обеспечивает корректное выполнение операций удаления, добавления и обновления кортежей в многотабличных представлениях базы данных на основе использования операций, выраженных формулами (4)–(8). Архитектура СКоП (см. рис. 1) предполагает следующие подсистемы: Коммутатор, Парсер и Локальный словарь базы данных.

Парсер — это подсистема, которая обеспечивает получение метаданных об отношениях, прямо и косвенно вовлеченных в запрос на обновление представления, из словаря базы данных, и их сохранение в Локальном словаре.

Локальный словарь обеспечивает хранение следующих основных метаданных: имена отношений и атрибутов каждого отношения, логические выражения, накладываемые на значения атрибутов и др. При инициализации клиентского приложения СКоП загружает данные Локального словаря в оперативную память. После обновления многотабличного представления в базе данных обновляется соответствующая информация в локальном словаре.

Коммутатор — подсистема, которая получает запрос пользователя на обновление представления, формирует текст транзакции, выполняющей обновление набора кортежей в целевом отношении базы данных, и затем запускает сформированный текст транзакции на сервере. Коммутатор возвращает клиентскому приложению отклик сервера (результат выполнения операции обновления или код ошибки). Для формирования текста транзакции Коммутатор использует данные из Локального словаря. Транзакция реализует операции реляционной алгебры, выраженные формулами (4)–(8).

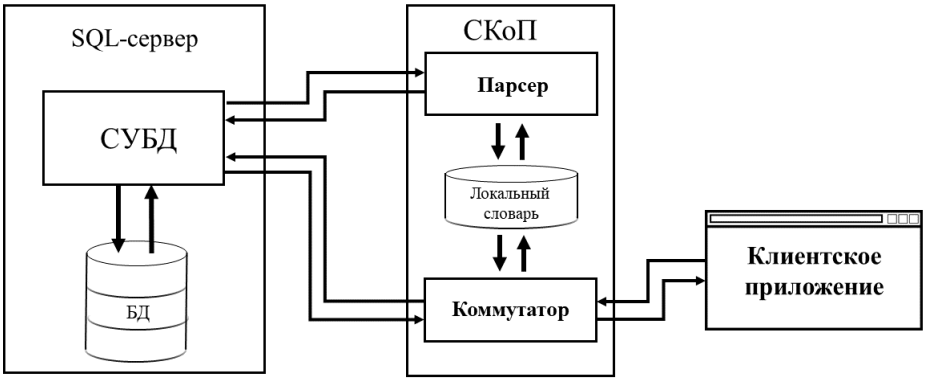

Рис. 1. Система баз данных с Сопроцессором коммутативных преобразований

3.2. Сопроцессор баз данных для PostgreSQL

Предложенная архитектура СКоП была реализована для свободной СУБД PostgreSQL [1], соответствующий Сопроцессор получил название pgCOPCT (PostgreSQL COProcessor Commutative Transformations). Сопроцессор разработан на программной платформе .NET Framework на языке программирования C# и представляет собой приложение ОС Windows. Реализация СКоП для другой реляционной СУБД потребует хотя и значительных, но относительно механических модификаций исходных текстов в соответствии с интерфейсами новой СУБД.

Работа сопроцессора pgCOPCT может быть кратко описана следующим образом (см. рис. 2). База данных под управлением СУБД PostgreSQL располагается на сервере. На клиентском компьютере устанавливается сопроцессор pgCOPCT и драйвер Npgsql для подключения к СУБД. В течение сеанса работы на стороне клиента в Локальном словаре сохраняются только данные, необходимые для обновления представлений. pgCOPCT состоит из следующих компонентов. Класс FormBuilder представляет собой конструктор запросов и обеспечивает формирование многотабличного представления. Класс DBConnector обеспечивает заполнение Локального словаря базы данных. Класс FormResult предоставляет оконный интерфейс редактирования многотабличных представлений. Класс DBEditor выполняет запуск сформированной транзакции на сервере.

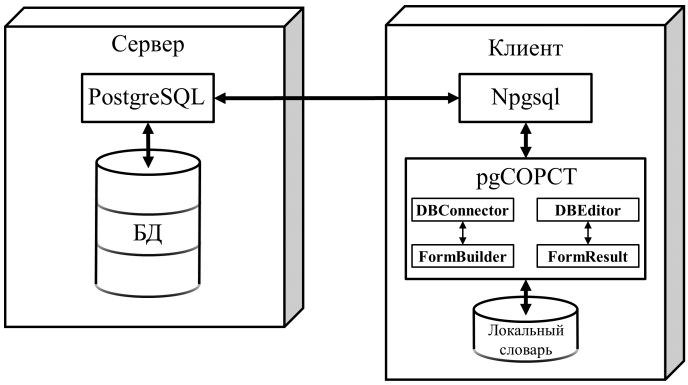

Рис. 2. Диаграмма развертывания сопроцессора pgCOPCT

4. Экспериментальное исследование

4.1. Цели и вычислительная среда экспериментов

В данном разделе описаны вычислительные эксперименты, исследующие эффективность предложенного подхода. *Эффективность* понимается как время выполнения операций обновления многотабличного представления с помощью сопроцессора коммутативных преобразований в сравнении с современными СУБД PostgreSQL, Oracle и MS SQL Server, которые для тех же целей используют триггеры.

Применение СКоП исследовалось для двух классов приложений: OLAP и OLTP. Приложения *OLAP (Online Analitical Processing, оперативный анализ данных)* связаны с выполнением сложных запросов на выборку данных из нескольких таблиц хранилища данных и использованием агрегационных функций. В связи с этим для приложений OLAP применение СКоП ограничивается случаем обновления хранилища данных, которому соответствует выполнение операций вставки новых кортежей в многотабличное представление. *Приложения OLTP (Online Transaction Processing, оперативная обработка транзакций)* подразумевают обработку коротких транзакций, которые выполняют вставку, удаление и обновление кортежей в базе данных в реальном времени. В соответствии с этим для приложений OLTP исследовалась эффективность СКоП при выполнении операций вставки, удаления и обновления кортежей в многотабличном представлении.

Эксперименты проводились с использованием синтетических баз данных, созданных в соответствии со спецификациями стандартных тестов консорциума TPC (Transaction Processing Council): TPC-H [10] для приложений класса OLAP и TPC-E [17] для приложений класса OLTP. При этом распределение кортежей в базе данных осуществлялось по правилу Зипфа «80-20» [7]: 80 % кортежей в целевом отношении соответствует 20 % кортежей в отношении-справочнике.

Аппаратная платформа экспериментов резюмирована ниже в таблице.

Таблица

Аппаратная платформа экспериментов

Характеристика	Значение
Процессор	Intel Core 2 Duo P7450 (2 ядра по 2,13 ГГц)
Оперативная память	4 Гб (DDR3-533)
Дисковая память	256 Гб (твердотельный накопитель OCZ)

4.2. Эффективность СКоП в приложениях класса OLAP

В стандартном тесте TPC-H [10] СУБД имитирует обработку заказов, используя базу данных, схема которой представлена на рис. 3. Базовыми отношениями многотабличного представления выбраны отношения CUSTOMER, ORDERS, LINEITEM и PARTSUPP. В качестве целевого отношения выбрана отношения LINEITEM. Многотабличное представление задействует все атрибуты целевого отношения, не являющиеся атрибутами внешнего ключа, а также атрибуты отношения CUSTOMER.Name, ORDERS.OrderPriority, PARTSUPP.Comment, к которым применено ограничение по атрибуту CUSTOMER.Address (отбор клиентов из специфицированного города). Варьируемым параметром экспериментов

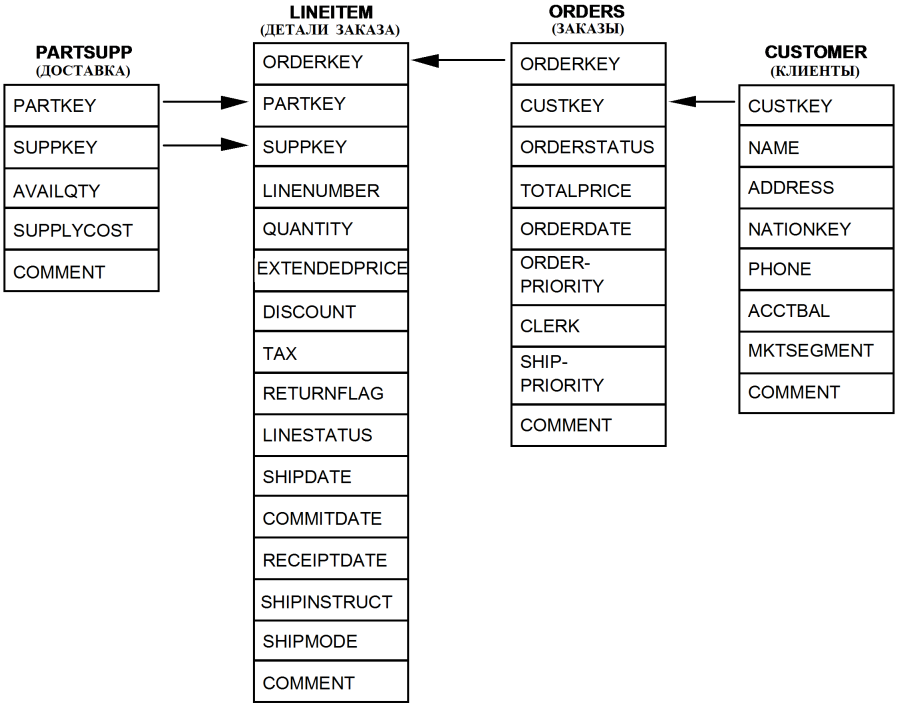

Рис. 3. Схема базы данных теста TPC-H

является количество кортежей в целевом отношении, соответствующих изменяемому кортежу в представлении.

Результаты экспериментов представлены на рис. 4. Можно видеть, что СУБД MS SQL Server, PostgreSQL и Oracle выполняют операции вставки данных в многотабличное представление с помощью СКоП быстрее, чем с помощью триггеров, на 10–35 %.

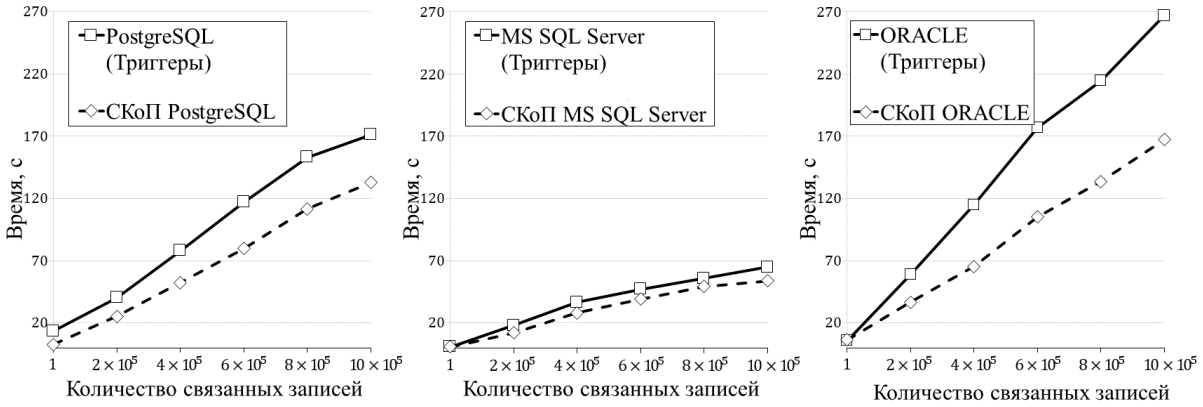

Рис. 4. Эффективность СКоП в приложениях класса OLAP

4.3. Эффективность СКоП в приложениях класса OLTP

В стандартном тесте TPC-E [17] СУБД имитирует торговлю на фондовой бирже и включает в себя следующие три сценария выполнения транзакций: обновление информации о сделке (Trade-Update), очистка информации об специфицированной сделке (Trade-Cleanup) и Market-Feed (протоколирование текущей рыночной активности), —

в каждом из которых выполняется модификация от четырех до шести отношений. Эксперименты проводились с использованием рассмотренного выше сопроцессора pgCOPCT.

Для каждого из указанных сценариев теста TPC-E было сформировано многотабличное представление, обновление которого выливается в выполнение транзакции, которая выполняет обновление отношений базы данных в соответствии со сценарием. Выполнение каждого сценария с использованием СКоП осуществлялось в режимах холодного и горячего запуска. Холодному запуску СКоП соответствует ситуация, когда Парсер сначала формирует Локальный словарь базы данных, а затем Коммутатор выполняет необходимые действия. Горячий запуск СКоП означает, что необходимость формирования Локального словаря базы данных отсутствует, и Коммутатор сразу выполняет необходимые действия.

Результаты экспериментов представлены на рис. 5. Можно видеть, что для каждого из рассмотренных сценариев в режиме горячего запуска pgCOPCT выполняет обновление многотабличного представления быстрее, чем СУБД PostgreSQL с помощью триггеров. Однако использование триггеров выгоднее в режиме холодного запуска. Таким образом, накладные расходы на поддержку Локального словаря базы данных являются необходимой платой за эффективность СКоП.

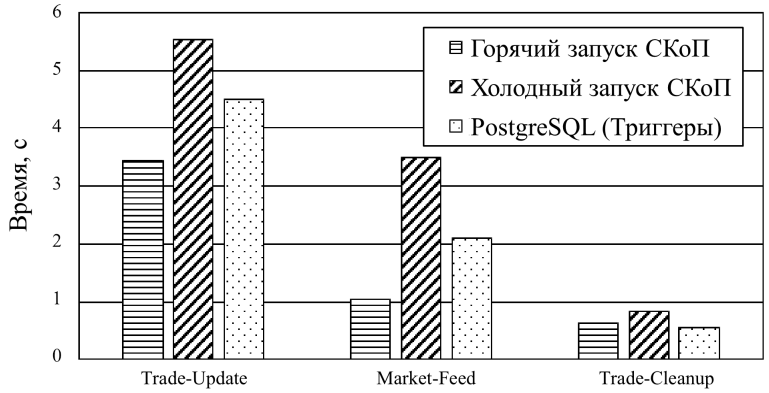

Рис. 5. Эффективность СКоП в приложениях класса OLTP

Заключение

В статье представлен подход к решению задачи корректного обновления многотабличных представлений (views) в реляционных базах данных на основе использования аппарата коммутативных преобразований.

Описан Сопроцессор СУБД, который размещается на клиентском компьютере и обеспечивает коммутативные преобразования в отношениях базы данных, хранимых на сервере. Сопроцессор состоит из трех подсистем: Парсер, Локальный словарь базы данных и Коммутатор. Парсер обеспечивает чтение метаданных словаря баз данных и их сохранение в Локальном словаре. Коммутатор, используя метаданные Локального словаря, формирует текст транзакции, реализующей коммутативные преобразования, и осуществляет запуск этой транзакции на сервере. Представлена реализация сопроцессора для свободной СУБД PostgreSQL.

Проведены вычислительные эксперименты, исследующие эффективность использования предложенного подхода в приложениях классов OLAP и OLTP с

использованием синтетических баз данных, специфицированных в стандартных тестах TPC-H и TPC-E соответственно. При обновлении многотабличных представлений Сопроцессор коммутативных преобразований показывает лучшее быстродействие, чем триггеры СУБД.

Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

Литература

1. Зыкин В.С. Редактор многотабличного представления данных: свидетельство о государственной регистрации программ для ЭВМ – № 2018661249 от 04.09.2018; Правообладатель: Омский государственный технический университет.
2. Зыкин В.С. Ссылочная целостность данных в корпоративных информационных системах // Информатика и ее применения. 2015. Т. 9. № 3. С. 119–127.
3. Зыкин С.В., Зыкин В.С. Коммутативные преобразования в базе данных при редактировании многотабличных запросов // Информационные технологии. 2018. Т. 24, № 5. С. 330–338. DOI: 10.17587/it.24.330-338.
4. Bancilhon F., Spyratos N. Update Semantics of Relational Views // ACM Trans. Database Syst. 1981. Vol. 6, No. 4. P. 557–575. DOI: 10.1145/319628.319634.
5. Bertossi L., Salimi B. Causes for Query Answers from Databases: Datalog Abduction, View-updates, and Integrity Constraints // Int. J. Approx. Reason. 2017. Vol. 90. P. 226–252. DOI: 10.1016/j.ijar.2017.07.010.
6. Dayal U., Bernstein P.A. On the Correct Translation of Update Operations on Relational Views // ACM Trans. Database Syst. 1982. Vol. 7, No. 3. P. 381–416. DOI: 10.1145/319732.319740.
7. Garcia-Molina H., Ullman J.D., Widom J. Database System Implementation. Prentice Hall, 2000. 653 p.
8. Ghandeharizadeh S., Yap J. SQL Query to Trigger Translation: A Novel Transparent Consistency Technique for Cache Augmented SQL Systems // Proceedings of the 28th International Workshop on Database and Expert Systems Applications, DEXA 2017, August 28–31, 2017, Lyon, France. P. 37–41. DOI: 10.1109/DEXA.2017.24.
9. Gottlob G., Paolini P., Zicari R. Properties and Update Semantics of Consistent Views // ACM Trans. Database Syst. 1988. Vol. 13, No. 4. P. 486–524. DOI: 10.1145/49346.50068.
10. Hayamizu Y., Kawamichi R., Goda K., Kitsuregawa M. Benchmarking and Performance Analysis of Event Sequence Queries on Relational Database // Proceedings of the 10th TPC Technology Conference Performance Evaluation and Benchmarking for the Era of Artificial Intelligence, TPCTC, August 27–31, 2018, Rio de Janeiro, Brazil. P. 110–125. DOI: 10.1007/978-3-030-11404-6_9.
11. Interim Report: ANSI/X3/SPARC Study Group on Data Base Management Systems. FDT - Bulletin of ACM SIGMOD. 1975. Vol. 7, No. 2. P. 1–140.
12. ISO/IEC 9075:1987 Information Technology. Database Languages. SQL. Washington, 1987.
13. ISO/IEC 9075-11:2016 Information technology. Database languages. SQL. Part 11: Information and Definition Schemas (SQL/Schemata). Washington. 2016. 327 p.

14. Keller A. Algorithms for Translating View Updates to Database Updates for Views Involving Selections, Projections and Joins // Proceedings of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS'85, March 25–27, 1985, Portland, USA. ACM, 1985. P. 154–163. DOI: 10.1145/325405.325423.

15. Langerak R. View Updates in Relational Databases with an Independent Scheme // ACM Trans. Database Syst. 1990. Vol. 15, No. 1. P. 40–66. DOI: 10.1145/77643.77645.

16. Lechtenbörger J. The Impact of the Constant Complement Approach Towards View Updating // Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'03, June 9–11, 2003, San Diego, CA, USA. ACM, 2003. P. 49–55. DOI: 10.1145/773153.773159.

17. Li Y., Levine C. Extending TPC-E to Measure Availability in Database Systems // Proceedings of the 10th Technology Conference Measurement and Characterization, August 29 – September 3, 2011, Seattle, WA, USA. P. 111–122. DOI: 10.1007/978-3-642-32627-1_8.

18. Masunaga Y. A Relational Database View Update Translation Mechanism // Proceedings of the 10th International Conference on Very Large Data Bases, VLDB'84, August 27–31, 1984, Singapore. P. 309–320.

19. Mosin S.V., Zykin S.V. Truth Space Method for Caching Database Queries // Моделирование и анализ информационных систем. 2015. Т. 22. № 2. С. 248–258.

20. Stonebraker M. Triggers and Inference In Database Systems. On Knowledge Base Management Systems (Islamorada). 1985. P. 297–314.

Зыкин Владимир Сергеевич, аспирант, старший преподаватель, кафедра прикладной математики и фундаментальной информатики, Омский государственный технический университет (Омск, Российская Федерация)

Цымблер Михаил Леонидович, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190206

UPDATING OF MULTI-TABLE VIEWS BASED ON COMMUTATIVE DATABASE TRANSFORMATIONS

© 2019 V.S. Zykin[1], M.L. Zymbler[2]

[1] *Omsk State Technical University (pr. Mira 11, Omsk, 644050 Russia),*
[2] *South Ural University (pr. Lenina 76, Chelyabinsk, 454080 Russia)*
E-mail: vszykin@mail.ru, mzym@susu.ru

Received: 26.09.2018

In modern relational database technologies, views implement the external layer of the ANSI–SPARC architecture, which encapsulates details of the database conceptual structure from end-users. However, when using views, we need to solve the problem of correct view updating: DBMS must execute insertion, deletion, and updating tuples of the view while providing correct modifications of corresponding target relation(s) of this view. To solve this problem, the SQL standard introduces a strict restriction: only one tuple in the target relation can correspond to the modified tuple in the view. Also, triggers are not a satisfactory solution of this problem because

of necessity of such a trigger for each view of the database, and unpredictable sequence in execution of triggers that belong to the same view, etc. The paper presents an approach to solve the problem of correct view updating based on the commutative database transformations. This does not limit the tuple uniqueness in the target relation that corresponds to the updated tuple in the view. We describe the DBMS Coprocessor, which is deployed on the client computer and provides commutative transformations in the database relations stored on the server side. The coprocessor generates a transaction's script that implements commutative transformations and runs the transaction on the server. We present implementation of the Coprocessor for the PostgreSQL open-source DBMS. Experimental evaluation confirms the effectiveness of the proposed approach in OLAP and OLTP applications.

Keywords: commutative transformation, relational algebra, multi-table view, view updating, relational DBMS, trigger.

FOR CITATION

Zykin V.S., Zymbler M.L. Updating Multi-table Views Based on Commutative Database Transformations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 2. pp. 92–106. (in Russian) DOI: 10.14529/cmse190206.

References

1. Zykin V.S. Multi-table Data View Editor: Certificate of State Registration of Computer Programs – No. 2018661249; registration date: 04.09.2018; Copyright holder: Omsk State Technical University.
2. Zykin V.S. Referential Integrity of Data in Corporate Information Systems. *Informatics and Applications.* 2015. vol. 9. no 3. pp. 119–127.
3. Zykin S.V., Zykin V.S. Commutative Conversion in the Database when Editing a Multitable Query. *Information Technologies.* 2018. vol. 24, no. 5. pp. 330–338. DOI: 10.17587/it.24.330-338.
4. Bancilhon F., Spyratos N. Update Semantics of Relational Views. *ACM Trans. Database Syst.* 1981. vol. 6, no. 4. pp. 557–575. DOI: 10.1145/319628.319634.
5. Bertossi L., Salimi B. Causes for Query Answers from Databases: Datalog Abduction, View-updates, and Integrity Constraints. *Int. J. Approx. Reason.* 2017. vol. 90. pp. 226–252. DOI: 10.1016/j.ijar.2017.07.010.
6. Dayal U., Bernstein P.A. On the Correct Translation of Update Operations on Relational Views. *ACM Trans. Database Syst.* 1982. vol. 7, no. 3. pp. 381–416. DOI: 10.1145/319732.319740.
7. Garcia-Molina H., Ullman J.D., Widom J. Database System Implementation. Prentice Hall, 2000. 653 p.
8. Ghandeharizadeh S., Yap J. SQL Query to Trigger Translation: A Novel Transparent Consistency Technique for Cache Augmented SQL Systems. Proceedings of the 28th International Workshop on Database and Expert Systems Applications, DEXA 2017, August 28–31, 2017, Lyon, France. pp. 37–41. DOI: 10.1109/DEXA.2017.24.
9. Gottlob G., Paolini P., Zicari R. Properties and Update Semantics of Consistent Views. *ACM Trans. Database Syst.* 1988. vol. 13, no. 4. p. 486–524. DOI: 10.1145/49346.50068.

10. Hayamizu Y., Kawamichi R., Goda K., Kitsuregawa M. Benchmarking and Performance Analysis of Event Sequence Queries on Relational Database. Proceedings of the 10th (TPC) Technology Conference Performance Evaluation and Benchmarking for the Era of Artificial Intelligence, TPCTC, August 27–31, 2018, Rio de Janeiro, Brazil. pp. 110–125. DOI: 10.1007/978-3-030-11404-6_9.

11. Interim Report: ANSI/X3/SPARC Study Group on Data Base Management Systems. FDT - Bulletin of ACM SIGMOD. 1975. vol. 7, no. 2. pp. 1–140.

12. ISO/IEC 9075:1987 Information technology. Database languages. SQL. Washington. 1987.

13. ISO/IEC 9075-11:2016 Information technology. Database languages. SQL. Part 11: Information and Definition Schemas (SQL/Schemata). Washington. 2016. 327 p.

14. Keller A. Algorithms for Translating View Updates to Database Updates for Views Involving Selections, Projections and Joins. Proceedings of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS'85, March 25–27, 1985, Portland, USA. ACM, 1985. pp. 154–163. DOI: 10.1145/325405.325423.

15. Langerak R. View Updates in Relational Databases with an Independent Scheme. *ACM Trans. Database Syst.* 1990. vol. 15, no. 1. pp. 40–66. DOI: 10.1145/77643.77645.

16. Lechtenbörger J. The Impact of the Constant Complement Approach Towards View Updating. Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'03, June 9–11, 2003, San Diego, CA, USA. ACM, 2003. pp. 49–55. DOI: 10.1145/773153.773159.

17. Li Y., Levine C. Extending TPC-E to Measure Availability in Database Systems. Proceedings of the 10th Technology Conference Measurement and Characterization, August 29 – September 3, 2011, Seattle, WA, USA. pp. 111–122. DOI: 10.1007/978-3-642-32627-1_8.

18. Masunaga Y. A Relational Database View Update Translation Mechanism. Proceedings of the 10th International Conference on Very Large Data Bases, VLDB'84, August 27–31, 1984, Singapore. pp. 309–320.

19. Mosin S.V., Zykin S.V. Truth Space Method for Caching Database Queries. *Modeling and Analysis of Information Systems*. 2015. vol. 22, no 2. pp. 248–258.

20. Stonebraker M. Triggers and Inference In Database Systems. On Knowledge Base Management Systems (Islamorada). 1985. pp. 297–314.